

AAAAAAA NNN NNN AAAAAAAA LLL YYY YYY ZZZZZZZZZZZZZZZZ  
AAAAAAA NNN NNN AAAAAAAA LLL YYY YYY ZZZZZZZZZZZZZZZZ  
AAAAAAA NNN NNN AAAAAAAA LLL YYY YYY ZZZZZZZZZZZZZZZZ  
AAA AAA NNN NNN AAA AAA LLL YYY YYY ZZZ  
AAA AAA NNN NNN AAA AAA LLL YYY YYY ZZZ  
AAA AAA NNN NNN AAA AAA LLL YYY YYY ZZZ  
AAA AAA NNNNNN NNN AAA AAA LLL YYY YYY ZZZ  
AAA AAA NNNNNN NNN AAA AAA LLL YYY YYY ZZZ  
AAA AAA NNNNNN NNN AAA AAA LLL YYY YYY ZZZ  
AAA AAA NNN NNN NNN AAA AAA LLL YYY ZZZ  
AAA AAA NNN NNN NNN AAA AAA LLL YYY ZZZ  
AAA AAA NNN NNN NNN AAA AAA LLL YYY ZZZ  
AAAAAAA NNN NNNNNN AAAAAAAA LLL YYY ZZZ  
AAAAAAA NNN NNNNNN AAAAAAAA LLL YYY ZZZ  
AAAAAAA NNN NNNNNN AAAAAAAA LLL YYY ZZZ  
AAA AAA NNN NNN AAA AAA LLL YYY ZZZ  
AAA AAA NNN NNN AAA AAA LLL YYY ZZZ  
AAA AAA NNN NNN AAA AAA LLL YYY ZZZ  
AAA AAA NNN NNN AAA AAA LLLL LLLL YYY ZZZZZZZZZZZZZZZZ  
AAA AAA NNN NNN AAA AAA LLLL LLLL YYY ZZZZZZZZZZZZZZZZ  
AAA AAA NNN NNN AAA AAA LLLL LLLL YYY ZZZZZZZZZZZZZZZZ

RRRRRRRR	MM	MM	SSSSSSSS
RRRRRRRR	MM	MM	SSSSSSSS
RR RR	MMMM	MMMM	SS
RR RR	MMMM	MMMM	SS
RR RR	MM MM	MM MM	SS
RR RR	MM MM	MM MM	SS
RRRRRRRR	MM	MM	SSSSSS
RRRRRRRR	MM	MM	SSSSSS
RR RR	MM	MM	SS
RR RR	MM	MM	SS
RR RR	MM	MM	SS
RR RR	MM	MM	SS
RR RR	MM	MM	SSSSSSSS
RR RR	MM	MM	SSSSSSSS

LL		SSSSSSSS
LL		SSSSSSSS
LL		SS
LL		SSSSSS
LL		SSSSSS
LL		SS
LLLLLLLL		SSSSSSSS
LLLLLLLL		SSSSSSSS

```
1 0001 0 Ztitle 'RMS - Main Module for ANALYZE/RMS_FILE'
2 0002 0 module rms      (main=anl$rms
3 0003 1 ident='V04-000') = begin
4
5 0005 1
6 0006 1 ****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 ****
28 0028 1 *
29 0029 1 *
30 0030 1 ++
31 0031 1 Facility: VAX/VMS Analyze Facility, Main Module for ANALYZE/RMS_FILE
32 0032 1
33 0033 1 Abstract: This is the main module for the ANALYZE/RMS_FILE command.
34 0034 1 It contains the routine that determines which mode we are
35 0035 1 to operate in. It also contains routines that don't fit
36 0036 1 anywhere else.
37 0037 1
38 0038 1
39 0039 1 Environment:
40 0040 1
41 0041 1 Author: Paul C. Anagnostopoulos, Creation Date: 18 February 1981
42 0042 1
43 0043 1 Modified By:
44 0044 1
45 0045 1 V03-003 DGB0046 Donald G. Blair 08-May-1984
46 0046 1 Fix condition handling for ANALYZRMS so that it
47 0047 1 returns status correctly upon image exit. Rather
48 0048 1 than always return anlrms$ notok, return the first
49 0049 1 error that occurs of the highest severity.
50 0050 1
51 0051 1 V03-002 PCA1011 Paul C. Anagnostopoulos 1-Apr-1983
52 0052 1 Change the message prefix to ANLRMSS$ to ensure that
53 0053 1 message symbols are unique across all ANALYZEs. This
54 0054 1 is necessitated by the new merged message files.
55 0055 1
56 0056 1 V03-001 PCA1002 Paul C. Anagnostopoulos 25-Oct-1982
57 0057 1 Add the ANL$PREPARE_QUOTED_STRING routine to format a
```

RMS  
V04-000

RMS - Main Module for ANALYZE/RMSFILE

H 6  
15-Sep-1984 23:52:21  
14-Sep-1984 11:52:58

VAX-11 Bliss-32 v4.0-742  
DISK\$VMSMASTER:[ANALYZ.SRC]RMS.B32;1

Page 2  
(1)

58  
59  
60

0058 1 |  
0059 1 |  
0060 1 |--

quoted string for inclusion in an FDL specification.  
Add code for /SUMMARY mode.

```

62 0061 1 %sbttl "Module Declarations"
63 0062 1
64 0063 1 : Libraries and Requires:
65 0064 1
66 0065 1
67 0066 1 library 'lib';
68 0067 1 require 'rmsreq';
69 0576 1
70 0577 1
71 0578 1 : Table of Contents:
72 0579 1
73 0580 1
74 0581 1 forward routine
75 0582 1     anl$rms: novalue,
76 0583 1     anl$unwind_handler,
77 0584 1     anl$worst_error_handler,
78 0585 1     anl$internalize_number,
79 0586 1     anl$check_flags: novalue,
80 0587 1     anl$prepare_quoted_string: novalue;
81 0588 1
82 0589 1
83 0590 1 : External References:
84 0591 1
85 0592 1
86 0593 1 external routine
87 0594 1     anl$check_mode,
88 0595 1     anl$fdl_mode,
89 0596 1     anl$format_error,
90 0597 1     anl$interactive_mode,
91 0598 1     cli$present: addressing_mode(general),
92 0599 1     lib$establish: addressing_mode(general),
93 0600 1     ots$cvt_til: addressing_mode(general),
94 0601 1     ots$cvt_tz_l: addressing_mode(general);
95 0602 1
96 0603 1
97 0604 1 : Global Variables:
98 0605 1
99 0606 1
100 0607 1 global
101 0608 1     anl$gb_mode: byte.      ! current mode of operation
102 0609 1     anl$worst_error:    ! this contains either success status or if
103 0610 1     initial$anlrms$ok);  ! errors occurred, it contains the first error
104 0611 1                           ! of the worst severity that occurred.
105 0612 1
106 0613 1
107 0614 1 : Own Variables:
108 0615 1
109 0616 1

```

```
111 0617 1 Isbttl 'ANL$RMS - Main Routine for ANALYZE/RMS FILE'
112 0618 1 ++
113 0619 1 Functional Description:
114 0620 1 This is the main routine, entered when the user performs an
115 0621 1 ANALYZE/RMS FILE command. We decide which mode of operation
116 0622 1 has been requested and do it.
117 0623 1
118 0624 1 Formal Parameters:
119 0625 1 none
120 0626 1
121 0627 1 Implicit Inputs:
122 0628 1 global data
123 0629 1
124 0630 1 Implicit Outputs:
125 0631 1 global data
126 0632 1
127 0633 1 Returned Value:
128 0634 1 none
129 0635 1
130 0636 1 Side Effects:
131 0637 1
132 0638 1 --+
133 0639 1
134 0640 2 global routine anl$rms: novalue = begin
135 0641 2
136 0642 2 lib$establish(anl$worst_error_handler);
137 0643 2
138 0644 2 : See which mode the user has requested. The default is /CHECK.
139 0645 2
140 0646 2 if cli$present(describe('FDL')) then (
141 0647 3 anl$gb_mode = anl$k_fdl;
142 0648 3 anl$fdl_mode();
143 0649 3
144 0650 3
145 0651 3 else if cli$present(describe('INTERACTIVE')) then (
146 0652 3 anl$gb_mode = anl$k_interactive;
147 0653 3 anl$interactive_mode();
148 0654 3
149 0655 3 else if cli$present(describe('STATISTICS')) then (
150 0656 3 anl$gb_mode = anl$k_statistics;
151 0657 3 anl$check_mode();
152 0658 3
153 0659 3 else if cli$present(describe('SUMMARY')) then (
154 0660 3 anl$gb_mode = anl$k_summary;
155 0661 3 anl$check_mode();
156 0662 3
157 0663 3 else (
158 0664 3 anl$gb_mode = anl$k_check;
159 0665 3 anl$check_mode();
160 0666 3
161 0667 3
162 0668 2 : if it was an interactive session, always return success. otherwise
163 0669 2 : return worst error
164 0670 2
165 0671 2 if .anl$gb_mode eq1 anl$k_interactive then
166 0672 2      $exit(code=anlrms$ok)
167 0673 2 else
```

RMS  
V04-000

RMS - Main Module for ANALYZE/RMS\_FILE  
ANL\$RMS - Main Routine for ANALYZE/RMS\_FILE

K 6  
15-Sep-1984 23:52:21  
14-Sep-1984 11:52:58

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[ANALYZ.SRC]RMS.B32;1

Page 5  
(3)

168 0674 2 Sexit(code=.anl\$worst\_error or sts\$inhib\_msg);  
169 0675 2  
170 0676 1 end;

.TITLE RMS RMS - Main Module for ANALYZE/RMS\_FILE  
.IDENT \V04-000\

.PSECT SPLIT\$,NOWRT,NOEXE,2

45 56 49 54 43 41 52 45 54 4E 49	00000 0000 P.AAB:	.ASCII \FDL\
	00000003 00004 P.AAA:	.BLKB 1
	00000000 00008 P.AAD:	.LONG 3
	0000000B 00018 P.AAC:	.ADDRESS P.AAB
	00000000 0001C P.AAF:	.ASCII \INTERACTIVE\
53 43 49 54 53 49 54 41 54 53	00020 P.AAF:	.BLKB 1
	0000000A 0002C P.AAE:	.LONG 11
	00000000 00030 P.AAH:	.ADDRESS P.AAD
59 52 41 4D 4D 55 53	00034 P.AAH:	.ASCII \STATISTICS\
	00000007 0003C P.AAG:	.BLKB 2
	00000000 00040 P.AAH:	.LONG 10
		.ADDRESS P.AAF
		.ASCII \SUMMARY\
		.BLKB 1
		.LONG 7
		.ADDRESS P.AAH

.PSECT SGLOBAL\$,NOEXE,2

00000 ANL\$GB_MODE::	.BLKB 1
00001	.BLKB 3
00000000G 00004 ANL\$WORST_ERROR::	.LONG ANLRMSS_OK
.EXTRN ANLRMSS_OK, ANLRMSS_ALLOC	
.EXTRN ANLRMSS_ANYTHING	
.EXTRN ANLRMSS_BACKUP, ANLRMSS_BKT	
.EXTRN ANLRMSS_BKTAREA	
.EXTRN ANLRMSS_BKTCHECK	
.EXTRN ANLRMSS_BKTFLAGS	
.EXTRN ANLRMSS_BKTFREE	
.EXTRN ANLRMSS_BKTKEY, ANLRMSS_BKITLEVEL	
.EXTRN ANLRMSS_BKTNEXT	
.EXTRN ANLRMSS_BKTPTRSIZE	
.EXTRN ANLRMSS_BKTRECID	
.EXTRN ANLRMSS_BKTRECID3	
.EXTRN ANLRMSS_BKTSAMPLE	
.EXTRN ANLRMSS_BKTVBNFREE	
.EXTRN ANLRMSS_BUCKETSIZE	
.EXTRN ANLRMSS_CELL, ANLRMSS_CELLDATA	
.EXTRN ANLRMSS_CELLFLAGS	
.EXTRN ANLRMSS_CHECKHDG	
.EXTRN ANLRMSS_CONTIG, ANLRMSS_CREATION	
.EXTRN ANLRMSS_CTLSIZE	
.EXTRN ANLRMSS_DATAREC	
.EXTRN ANLRMSS_DATABKTVBN	

.EXTRN ANLRMSS\_DUMPHEADING  
.EXTRN ANLRMSS\_EOF, ANLRMSS\_ERRORCOUNT  
.EXTRN ANLRMSS\_ERRORNONE  
.EXTRN ANLRMSS\_ERRORS, ANLRMSS\_EXPIRATION  
.EXTRN ANLRMSS\_FILEATTR  
.EXTRN ANLRMSS\_FILEHDR  
.EXTRN ANLRMSS\_FILEID, ANLRMSS\_FILEORG  
.EXTRN ANLRMSS\_FILESPEC  
.EXTRN ANLRMSS\_FLAG, ANLRMSS\_GLOBALBUFS  
.EXTRN ANLRMSS\_HEXDATA  
.EXTRN ANLRMSS\_HEXHEADING1  
.EXTRN ANLRMSS\_HEXHEADING2  
.EXTRN ANLRMSS\_IDXAREA  
.EXTRN ANLRMSS\_IDXAREAALLOC  
.EXTRN ANLRMSS\_IDXAREABKTSZ  
.EXTRN ANLRMSS\_IDXAREANEXT  
.EXTRN ANLRMSS\_IDXAREANOALLOC  
.EXTRN ANLRMSS\_IDXAREAQTY  
.EXTRN ANLRMSS\_IDXAREARECL  
.EXTRN ANLRMSS\_IDXAREAUSED  
.EXTRN ANLRMSS\_IDXKEY, ANLRMSS\_IDXKEYAREAS  
.EXTRN ANLRMSS\_IDXKEYBKTSZ  
.EXTRN ANLRMSS\_IDXKEYBYTES  
.EXTRN ANLRMSS\_IDXKEY1TYPE  
.EXTRN ANLRMSS\_IDXKEYDATAVBN  
.EXTRN ANLRMSS\_IDXKEYFILL  
.EXTRN ANLRMSS\_IDXKEYFLAGS  
.EXTRN ANLRMSS\_IDXKEYKEYSZ  
.EXTRN ANLRMSS\_IDXKEYNAME  
.EXTRN ANLRMSS\_IDXKEYNEXT  
.EXTRN ANLRMSS\_IDXKEYMINREC  
.EXTRN ANLRMSS\_IDXKEYNULL  
.EXTRN ANLRMSS\_IDXKEYPOSS  
.EXTRN ANLRMSS\_IDXKEYROOTLVL  
.EXTRN ANLRMSS\_IDXKEYROOTVBN  
.EXTRN ANLRMSS\_IDXKEYSEGS  
.EXTRN ANLRMSS\_IDXKEYSIZES  
.EXTRN ANLRMSS\_IDXPRIMREC  
.EXTRN ANLRMSS\_IDXPRIMRECFLAGS  
.EXTRN ANLRMSS\_IDXPRIMRECID  
.EXTRN ANLRMSS\_IDXPRIMRECLEN  
.EXTRN ANLRMSS\_IDXPRIMRECRRV  
.EXTRN ANLRMSS\_IDXPROAREAS  
.EXTRN ANLRMSS\_IDXPROLOG  
.EXTRN ANLRMSS\_IDXREC, ANLRMSS\_IDXRECPTR  
.EXTRN ANLRMSS\_IDXSIDR  
.EXTRN ANLRMSS\_IDXSIDRDUPCNT  
.EXTRN ANLRMSS\_IDXSIDRFLAGS  
.EXTRN ANLRMSS\_IDXSIDRRECID  
.EXTRN ANLRMSS\_IDXSIDRPTRFLAGS  
.EXTRN ANLRMSS\_IDXSIDRPTRREF  
.EXTRN ANLRMSS\_INTERCOMMAND  
.EXTRN ANLRMSS\_INTERHDG  
.EXTRN ANLRMSS\_LONGREC  
.EXTRN ANLRMSS\_MAXRECSIZE  
.EXTRN ANLRMSS\_NOBACKUP  
.EXTRN ANLRMSS\_NOEXPIRATION

RMS  
V04-000

RMS - Main Module for ANALYZE/RMS\_FILE  
ANL\$RMS - Main Routine for ANALYZE/RMS\_FILE

M 6

15-Sep-1984 23:52:21  
14-Sep-1984 11:52:58

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[ANALYZ.SRC]RMS.B32:1

Page 7  
(3)

.EXTRN ANLRMSS\_NOSPANFILLER  
.EXTRN ANLRMSS\_PERFORM  
.EXTRN ANLRMSS\_PROLOGFLAGS  
.EXTRN ANLRMSS\_PROLOGVER  
.EXTRN ANLRMSS\_PROT, ANLRMSS\_RECATTR  
.EXTRN ANLRMSS\_RECFCMT, ANLRMSS\_RECLAIMBKT  
.EXTRN ANLRMSS\_RELBUCKET  
.EXTRN ANLRMSS\_RELEOFBN  
.EXTRN ANLRMSS\_RELMAXREC  
.EXTRN ANLRMSS\_RELPROLOG  
.EXTRN ANLRMSS\_RELIAB, ANLRMSS\_REVISION  
.EXTRN ANLRMSS\_STATHDG  
.EXTRN ANLRMSS\_SUMMARYHDG  
.EXTRN ANLRMSS\_OWNERUIC  
.EXTRN ANLRMSS\_JNL, ANLRMSS\_AIJNL  
.EXTRN ANLRMSS\_BIJNL, ANLRMSS\_ATJNL  
.EXTRN ANLRMSS\_ATTOP, ANLRMSS\_BADCMD  
.EXTRN ANLRMSS\_BADPATH  
.EXTRN ANLRMSS\_BADVBN, ANLRMSS\_DOWHELP  
.EXTRN ANLRMSS\_DOWNPATH  
.EXTRN ANLRMSS\_EMPTYBKT  
.EXTRN ANLRMSS\_NODATA, ANLRMSS\_NODOWN  
.EXTRN ANLRMSS\_NONEXT, ANLRMSS\_NORECLAIMED  
.EXTRN ANLRMSS\_NORECS, ANLRMSS\_NORRV  
.EXTRN ANLRMSS\_RESTDONE  
.EXTRN ANLRMSS\_STACKFULL  
.EXTRN ANLRMSS\_UNINITINDEX  
.EXTRN ANLRMSS\_FDLIDENT  
.EXTRN ANLRMSS\_FDLSYSTEM  
.EXTRN ANLRMSS\_FDLSOURCE  
.EXTRN ANLRMSS\_FDLFILE  
.EXTRN ANLRMSS\_FDLALLOC  
.EXTRN ANLRMSS\_FDLNOALLOC  
.EXTRN ANLRMSS\_FDLBESTTRY  
.EXTRN ANLRMSS\_FDLBUCKETSIZE  
.EXTRN ANLRMSS\_FDLCLUSTERSIZE  
.EXTRN ANLRMSS\_FDLCONTIG  
.EXTRN ANLRMSS\_FDLEXTENSION  
.EXTRN ANLRMSS\_FDLGLOBALBUFS  
.EXTRN ANLRMSS\_FDLMAXRECORD  
.EXTRN ANLRMSS\_FDLFILENAME  
.EXTRN ANLRMSS\_FDLORG, ANLRMSS\_FDLOWNER  
.EXTRN ANLRMSS\_FDLPROTECTION  
.EXTRN ANLRMSS\_FDLRECORD  
.EXTRN ANLRMSS\_FDLSPAN  
.EXTRN ANLRMSS\_FDLCC, ANLRMSS\_FDLVFCSIZE  
.EXTRN ANLRMSS\_FDLFORMAT  
.EXTRN ANLRMSS\_FDLSIZE  
.EXTRN ANLRMSS\_FDLAREA  
.EXTRN ANLRMSS\_FDLKEY, ANLRMSS\_FDLCHANGES  
.EXTRN ANLRMSS\_FDLDATAAREA  
.EXTRN ANLRMSS\_FDLDATAFILL  
.EXTRN ANLRMSS\_FDLDATAKEYCOMPB  
.EXTRN ANLRMSS\_FDLDATARECCOMPB  
.EXTRN ANLRMSS\_FDLDUPS  
.EXTRN ANLRMSS\_FDLINDEXAREA  
.EXTRN ANLRMSS\_FDLINDEXCOMPB

27  
RMS  
V04-000

RMS - Main Module for ANALYZE/RMS FILE  
ANL\$RMS - Main Routine for ANALYZE/RMS\_FILE

N 6

15-Sep-1984 23:52:21  
14-Sep-1984 11:52:58

VAX-11 Bliss-32 v4.0-742  
DISK\$VMSMASTER:[ANALYZ.SRC]RMS.B32;1

Page 8  
(3)

.EXTRN ANLRMSS\$\_FDLINDEXFILL  
.EXTRN ANLRMSS\$\_FDLINDEXAREA  
.EXTRN ANLRMSS\$\_FDLKEYNAME  
.EXTRN ANLRMSS\$\_FDLNORECS  
.EXTRN ANLRMSS\$\_FDLNULLKEY  
.EXTRN ANLRMSS\$\_FDLNULLVALUE  
.EXTRN ANLRMSS\$\_FDLPROLOG  
.EXTRN ANLRMSS\$\_FDLSEGLENGTH  
.EXTRN ANLRMSS\$\_FDLSEGPOS  
.EXTRN ANLRMSS\$\_FDLSEGTYPE  
.EXTRN ANLRMSS\$\_FDLANALAREA  
.EXTRN ANLRMSS\$\_FDLRECL  
.EXTRN ANLRMSS\$\_FDLANALKEY  
.EXTRN ANLRMSS\$\_FDLDATAKEYCOMP  
.EXTRN ANLRMSS\$\_FDLDATAARECCOMP  
.EXTRN ANLRMSS\$\_FDLDATARECS  
.EXTRN ANLRMSS\$\_FDLDATASPACE  
.EXTRN ANLRMSS\$\_FDLDEPTH  
.EXTRN ANLRMSS\$\_FDLDUPSPER  
.EXTRN ANLRMSS\$\_FDLIDXCOMP  
.EXTRN ANLRMSS\$\_FDLIDXFILL  
.EXTRN ANLRMSS\$\_FDLIDXSPACE  
.EXTRN ANLRMSS\$\_FDLIDX1RECS  
.EXTRN ANLRMSS\$\_FDLLENMEAN  
.EXTRN ANLRMSS\$\_FDLLENMEAN  
.EXTRN ANLRMSS\$\_STATAREA  
.EXTRN ANLRMSS\$\_STATRECL  
.EXTRN ANLRMSS\$\_STATKEY  
.EXTRN ANLRMSS\$\_STATDEPTH  
.EXTRN ANLRMSS\$\_STATIDX1RECS  
.EXTRN ANLRMSS\$\_STATIDXLENMEAN  
.EXTRN ANLRMSS\$\_STATIDXSPACE  
.EXTRN ANLRMSS\$\_STATIDXFILL  
.EXTRN ANLRMSS\$\_STATIDXCOMP  
.EXTRN ANLRMSS\$\_STATDATARECS  
.EXTRN ANLRMSS\$\_STATDUPSPER  
.EXTRN ANLRMSS\$\_STATDATALENMEAN  
.EXTRN ANLRMSS\$\_STATDATASPACE  
.EXTRN ANLRMSS\$\_STATDATAFILL  
.EXTRN ANLRMSS\$\_STATDATAKEYCOMP  
.EXTRN ANLRMSS\$\_STATDATAARECCOMP  
.EXT... ANLRMSS\$\_STATEFFICIENCY  
.EXTRN ANLRMSS\$\_BADAREA1ST2  
.EXTRN ANLRMSS\$\_BADAREABKTSIZE  
.EXTRN ANLRMSS\$\_BADAREAFIT  
.EXTRN ANLRMSS\$\_BADAREAID  
.EXTRN ANLRMSS\$\_BADAREANEXT  
.EXTRN ANLRMSS\$\_BADAREAROOT  
.EXTRN ANLRMSS\$\_BADAREAUSED  
.EXTRN ANLRMSS\$\_BADBKTAAREAID  
.EXTRN ANLRMSS\$\_BADBKTCHECK  
.EXTRN ANLRMSS\$\_BADBKTFREE  
.EXTRN ANLRMSS\$\_BADBKTKEYID  
.EXTRN ANLRMSS\$\_BADBKTLLEVEL  
.EXTRN ANLRMSS\$\_BADBKTROOTBIT  
.EXTRN ANLRMSS\$\_BADBKTSAMPLE  
.EXTRN ANLRMSS\$\_BADCELLFIT

.EXTRN ANLRMSS\$-BADCHECKSUM  
 .EXTRN ANLRMSS\$-BADDATARECBITS  
 .EXTRN ANLRMSS\$-BADDATARECFIT  
 .EXTRN ANLRMSS\$-BADDATARECPS  
 .EXTRN ANLRMSS\$-BAD3IDXKEYFIT  
 .EXTRN ANLRMSS\$-BADIDXLASTKEY  
 .EXTRN ANLRMSS\$-BADIDXORDER  
 .EXTRN ANLRMSS\$-BADIDXRECBITS  
 .EXTRN ANLRMSS\$-BADIDXRECFIT  
 .EXTRN ANLRMSS\$-BADIDXRECPS  
 .EXTRN ANLRMSS\$-BADKEYAREAID  
 .EXTRN ANLRMSS\$-BADKEYDATABKT  
 .EXTRN ANLRMSS\$-BADKEYDATAFIT  
 .EXTRN ANLRMSS\$-BADKEYDATATYPE  
 .EXTRN ANLRMSS\$-BADKEYIDXBKT  
 .EXTRN ANLRMSS\$-BADKEYFILL  
 .EXTRN ANLRMSS\$-BADKEYFIT  
 .EXTRN ANLRMSS\$-BADKEYREFID  
 .EXTRN ANLRMSS\$-BADKEYROOTLEVEL  
 .EXTRN ANLRMSS\$-BADKEYSEGCOUNT  
 .EXTRN ANLRMSS\$-BADKEYSEGVEC  
 .EXTRN ANLRMSS\$-BADKEYSUMMARY  
 .EXTRN ANLRMSS\$-BADREADNOPAR  
 .EXTRN ANLRMSS\$-BADREADPAR  
 .EXTRN ANLRMSS\$-BADSIDRDUPT  
 .EXTRN ANLRMSS\$-BADSIDRPTRFIT  
 .EXTRN ANLRMSS\$-BADSIDRPTRSZ  
 .EXTRN ANLRMSS\$-BADSIDRSIZE  
 .EXTRN ANLRMSS\$-BADSTREAMEOF  
 .EXTRN ANLRMSS\$-BADVBNFREE  
 .EXTRN ANLRMSS\$-BKTLOOP  
 .EXTRN ANLRMSS\$-EXTENDERERR  
 .EXTRN ANLRMSS\$-FLAGERROR  
 .EXTRN ANLRMSS\$-MISSINGBKT  
 .EXTRN ANLRMSS\$-NOTOK, ANLRMSS\$-SPANERROR  
 .EXTRN ANLRMSS\$-TOOMANYRECS  
 .EXTRN ANLRMSS\$-UNWIND, ANLRMSS\$-VFCTOOSHORT  
 .EXTRN ANLRMSS\$-CACHEFULL  
 .EXTRN ANLRMSS\$-CACHERELFAIL  
 .EXTRN ANLS\$CHECK\_MODE, ANLS\$FDL\_MODE  
 .EXTRN ANLS\$FORMAT\_ERROR  
 .EXTRN ANLS\$INTERACTIVE\_MODE  
 .EXTRN CLISPRESNT, LIB\$ESTABLISH  
 .EXTRN OTSSCVT TI\_L, OTSSCVT TZ\_L  
 .EXTRN SYS\$EXIT

.PSECT SCODES,NOWRT,2

53	0000	CF	9E	00002	.ENTRY	ANLSRMS, Save R2,R3	0641
52	00000000G	00	9E	00007	MOVAB	ANLS\$GB MODE, R3	
	0000V	CF	9F	0000E	MOVAB	CLISPRESNT, R2	0643
00000000G	00	0000	FB	00012	PUSHAB	ANLS\$WORST_ERROR_HANDLER	
	0000	CF	9F	00019	CALLS	#1, LIB\$ESTABLISH	0647
62	01	FB	0001D	PUSHAB	P.AAA		
0A	50	E9	00020	CALLS	#1, CLISPRESNT		
				BLBC	RO, 1\$		

0000G	63	02	90 00023	MOV B	#2, ANL\$GB_MODE	: 0648	
	CF	00	FB 00026	CALLS	#0, ANL\$FDC_MODE	: 0649	
		3A	11 0002B	BRB	6\$	: 0647	
		0000'	CF 9F 0002D	18:	PUSHAB	P.AAC	: 0651
	62	01	FB 00031	CALLS	#1, CLISPRES	ENT	
	0A	50	E9 00034	BLBC	RO, 2\$		
0000G	63	03	90 00037	MOV B	#3, ANL\$GB_MODE	: 0652	
	CF	00	FB 0003A	CALLS	#0, ANL\$INTERACTIVE_MODE	: 0653	
		26	11 0003F	BRB	6\$	: 0651	
		0000'	CF 9F 00041	28:	PUSHAB	P.AAE	: 0655
	62	01	FB 00045	CALLS	#1, CLISPRES	ENT	
	05	50	E9 00048	BLBC	RO, 3\$		
	63	04	90 0004B	MOV B	#4, ANL\$GB_MODE	: 0656	
		12	11 0004E	BRB	5\$	: 0657	
		0000'	CF 9F 00050	38:	PUSHAB	P.AAG	: 0659
	62	01	FB 00054	CALLS	#1, CLISPRES	ENT	
	05	50	E9 00057	BLBC	RO, 4\$		
	63	05	90 0005A	MOV B	#5, ANL\$GB_MODE	: 0660	
		03	11 0005D	BRB	5\$	: 0661	
0000G	63	01	90 0005F	48:	MOV B	#1, ANL\$GB_MODE	: 0664
	CF	00	FB 00062	58:	CALLS	#0, ANL\$CHECK_MODE	: 0665
	03	63	91 00067	68:	CMPB	ANL\$GB_MODE, 73	: 0671
		08	12 0006A	BNEQ	7\$		
		00000000G	8F DD 0006C		PUSHL	#ANLRMSS_OK	: 0672
		09	11 00072		BRB	8\$	
7E	04	A3 10000000	8F C9 00074	78:	BISL3	#268435456, ANL\$WORST_ERROR, -(SP)	: 0674
	00	00000000G	01 FB 0007D	88:	CALLS	#1, SYS\$EXIT	
			04 00084		RET		: 0676

: Routine Size: 133 bytes. Routine Base: SCODE\$ + 0000

```
172 0677 1 %sbtll 'ANL$UNWIND_HANDLER - Unwind to Caller'  
173 0678 1 !***  
174 0679 1  
175 0680 1 Functional Description:  
176 0681 1 This condition handler is established at various points  
177 0682 1 in analyzrms and allows the stack to be unwound  
178 0683 1 and execution continued after any of an assortment  
179 0684 1 of serious errors occurs.  
180 0685 1  
181 0686 1 Formal Parameters:  
182 0687 1 signal_args = Address of signal argument list  
183 0688 1 mechanism_args = Address of mechanism argument list  
184 0689 1  
185 0690 1 Implicit Inputs:  
186 0691 1 none  
187 0692 1  
188 0693 1 Returned Value:  
189 0694 1 ss$_resignal This was not an anlrms$_unwind condition.  
190 0695 1 ss$_continue  
191 0696 1  
192 0697 1 Side Effects:  
193 0698 1 anl$worst_error is updated with highest severity error.  
194 0699 1  
195 0700 1 !---  
196 0701 1  
197 0702 2 global routine anl$unwind_handler (signal_args, mechanism_args) = begin  
198 0703 2  
199 0704 2 map  
200 0705 2 signal_args: ref bblock, ! Address of signal argument list  
201 0706 2 mechanism_args: ref bblock; ! Address of mechanism argument list  
202 0707 2  
203 0708 2 local  
204 0709 2 code: bblock [long], ! Condition code (longword)  
205 0710 2 status:  
206 0711 2  
207 0712 2 code = .signal_args [chf$1_sig_name]; ! Get condition code  
208 0713 2  
209 0714 2 ! If the condition is not anlrms$unwind, then we resignal it.  
210 0715 2  
211 0716 2 if .code nequ anlrms$unwind then  
212 0717 2 return ss$_resignal  
213 0718 2  
214 0719 2 ! It's a drastic structure error. We can no longer continue what we  
215 0720 2 were doing. In interactive mode, we want to return to user command  
216 0721 2 level. In check mode, we want to quit analyzing this file and  
217 0722 2 go on to the next file. In FDL mode, since there can only be 1  
218 0723 2 file spec, we just quit. In all cases, we put out an error message  
219 0724 2 and back out by unwinding to the frame of the caller of the routine  
220 0725 2 that called lib$establish. Note that since we do not resignal  
221 0726 2 the error to allow anl$worst_error_handler to save the anl$worst_error,  
222 0727 2 ! we must save it here.  
223 0728 2  
224 0729 3 else {  
225 0730 3 if severity_level (.code) gtr  
226 0731 3 severity_level (.anl$worst_error) ! If higher than watermark  
227 0732 3 then anl$worst_error = .code; ! -then set new worst error  
228 0733 3
```

```

229 0734 3 anl$format_error(anlrms$_unwind);
230 0735 3 status=$unwind();
231 0736 3 check (.status..status);
232 0737 3
233 0738 3 return ss$_continue;
234 0739 2 );
235 0740 2
236 0741 1 end;

```

## .EXTRN SYSSUNWIND

			54 00000000G	001C 00000	.ENTRY	ANL\$UNWIND_HANDLER, Save R2,R3,R4	0702
			50 04	0F D0 00002	MOVL	#ANLRMSS UNWIND, R4	0712
			53 04	AC D0 00009	MOVL	SIGNAL_ARGS, R0	0716
			54	A0 D0 00000	MOVL	4(R0), CODE	0729
				53 D1 00011	CMPL	CODE, R4	
				06 13 00014	BEQL	1\$	
			50 0918	8F 3C 00016	MOVZWL	#2328, R0	
				04 0001B	RET		
51	50	50	50	53 D0 0001C	MOVL	CODE, TMP_CODE	0730
			03 00	EF 0001F	EXTZV	#0, #3, TMP_CODE, R1	
			01 00	EF 00024	EXTZV	#0, #1, TMP_CODE, R0	
			50 04	C4 00029	MULL2	#4, R0	
			51 50	C2 0002C	SUBL2	R0, R1	
			51 03	C0 0002F	ADDL2	#3, R1	
52	50	50	50	0000* CF	MOVL	ANL\$WORST_ERROR, TMP_CODE	0731
			03 00	EF 00037	EXTZV	#0, #3, TMP_CODE, R2	
			01 00	EF 0003C	EXTZV	#0, #1, TMP_CODE, R0	
			50 04	C4 00041	MULL2	#4, R0	
			52 50	C2 00044	SUBL2	R0, R2	
			50 03	A2 9E 00047	MOVAB	3(R2) R0	
			50	51 D1 0004B	CMPL	R1, R0	
				05 15 0004E	BLEQ	2\$	
			0000* CF	53 D0 00050	MOVL	CODE, ANL\$WORST_ERROR	0732
			0000G CF	54 DD 00055	PUSHL	R4	0734
				01 FB 00057	CALLS	#1, ANL\$FORMAT_ERROR	
			00000006 00	7E 7C 0005C	CLRO	-(SP)	0735
			09	02 FB 0005E	CALLS	#2, SYSSUNWIND	
			00000006 00	50 E8 00065	BLBS	STATUS, 3\$	0736
				50 DD 00068	PUSHL	STATUS	
			0000000G 00	01 FB 0006A	CALLS	#1, LIB\$SIGNAL	
			50	01 D0 00071	MOVL	#1, R0	0738
				04 00074	RET		0741

; Routine Size: 117 bytes. Routine Base: \$CODE\$ + 0085

```

: 238 1 %sbtll 'ANL$WORST_ERROR_HANDLER = Baddest error handler in the West'
: 239 1
: 240 1
: 241 1 Functional Description:
: 242 1 This condition handler is established by the main routine in
: 243 1 analyzrms. It gains control when any error is signaled except
: 244 1 for ANL$RM$ UNWIND, which is handled specially by the
: 245 1 ANL$UNWIND HANDLER. If the error signaled is more severe than any
: 246 1 which has preceded it, save the error status as the
: 247 1 anl$worst_error. The resignal the error so the last-chance
: 248 1 condition handler can get a crack at the error.
: 249 1
: 250 1 Formal Parameters:
: 251 1 signal_args = Address of signal argument list
: 252 1 mechanism_args = Address of mechanism array
: 253 1
: 254 1 Implicit Inputs:
: 255 1 none
: 256 1
: 257 1 Returned Value:
: 258 1 ss$_resignal Continue to search call frames.
: 259 1
: 260 1 Side Effects:
: 261 1 anl$worst_error is updated with highest severity error.
: 262 1
: 263 1
: 264 2 global routine anl$worst_error_handler (signal_args, mechanism_args) = begin
: 265 2
: 266 2 map
: 267 2   signal_args: ref bblock; ! Address of signal argument list
: 268 2   mechanism_args: ref bblock; ! Address of mechanism argument list
: 269 2
: 270 2 local
: 271 2   code: bblock [long]; ! Condition code (longword)
: 272 2
: 273 2   code = .signal_args [chf$1_sig_name]; ! Get condition code
: 274 2   if severity_level (.code) gtr
: 275 2     severity_level (.anl$worst_error) ! If higher than watermark
: 276 2   then anl$worst_error = .code; ! -then set new worst error
: 277 2
: 278 2   return ss$_resignal;
: 279 2
: 280 2 end;

```

						.ENTRY	ANL\$WORST_ERROR_HANDLER, Save R2,R3	
		50	04	000C 00000		MOVL	SIGNAL_ARGS, R0	: 0768
		53	04	AC D0 00002		MOVL	4(R0), CODE	: 0777
		50	04	A0 D0 00006		MOVL	CODE, TMP_CODE	: 0778
		03	00	00000		EXTZV	#0, #3, TMP_CODE, R1	
		01	00	EF 0000A		EXTZV	#0, #1, TMP_CODE, R0	
		50	04	00 EF 00012		MULL2	#4, R0	
		51	04	C4 00017		SUBL2	R0, R1	
		51	03	C2 0001A				
		51	03	C0 0001D		ADDL2	#3, R1	

RMS  
V04-000

RMS - Main Module for ANALYZE/RMSFILE  
ANL\$WORST\_ERROR\_HANDLER - Baddest error handler

6 7  
15-Sep-1984 14-Sep-1984 23:52:21 11:52:58

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[ANALYZ.SRC]RMS.B32;1

Page 14  
(5)

52	50	50	0000'	CF	00 00020	MOVL	ANL\$WORST_ERROR, TMP_CODE	0779
			03	00	EF 00025	EXTZV	#0, #3, TMP_CODE, R2	
			01	00	EF 0002A	EXTZV	#0, #1, TMP_CODE, R0	
			50	04	C4 0002F	MULL2	#4, R0	
			52	50	C2 00032	SUBL2	R0, R2	
			50	03	A2 9E 00035	MOVAB	3(R2) R0	
			50	51	D1 00039	CMPL	R1, R0	
			0000'	05	15 0003C	BLEQ	1S	
			50	0918	53 D0 0003E	MOVL	CODE, ANL\$WORST_ERROR	0780
				8F	3C 00043	MOVZWL	#2328, R0	0782
					18: 04 0004B	RET		0784

: Routine Size: 73 bytes. Routine Base: SCODES + 00FA

```
282 0785 1 Isbttl 'ANL$INTERNALIZE_NUMBER - Convert String to Longword'
283 0786 1 ++
284 0787 1 Functional Description:
285 0788 1 This routine will convert the ASCII representation of a decimal
286 0789 1 or hexadecimal number to a longword. It is here for lack of a
287 0790 1 better place.
288 0791 1
289 0792 1 Formal Parameters:
290 0793 1     string      Address of descriptor of ASCII number. Decimal
291 0794 1     numbers are just digits, while hexadecimal numbers
292 0795 1     begin with %X or are enclosed in %X'...'.
293 0796 1     longword    Address of longword in which to return value.
294 0797 1
295 0798 1 Implicit Inputs:
296 0799 1     global data
297 0800 1
298 0801 1 Implicit Outputs:
299 0802 1     global data
300 0803 1
301 0804 1 Returned Value:
302 0805 1     True if number was valid, false if invalid.
303 0806 1
304 0807 1 Side Effects:
305 0808 1
306 0809 1     --
307 0810 1
308 0811 1
309 0812 2 global routine anl$internalize_number(string,longword) = begin
310 0813 2
311 0814 2     bind
312 0815 2     string_dsc = .string: descriptor;
313 0816 2
314 0817 2     local
315 0818 2     status: long,
316 0819 2     sp: ref block[byte],
317 0820 2     hex_dsc: descriptor;
318 0821 2
319 0822 2
320 0823 2     ! If the string is null, then it's invalid.
321 0824 2
322 0825 2     if .string_dsc[len] equ 0 then
323 0826 2         return false;
324 0827 2
325 0828 2     ! Split up depending upon whether it's a decimal or hex number.
326 0829 2
327 0830 2     if ch$eq1(minul(.string_dsc[len],2),.string_dsc[ptr], 2,uplit byte('ZX'), ' ') then (
328 0831 2
329 0832 2         ! We have a hex number. Build a descriptor of the actual digits.
330 0833 2         ! If the third character is an apostrophe, then we must find the
331 0834 2         ! matching apostrophe.
332 0835 2
333 0836 2         if ch$rchar(.string_dsc[ptr]+2) equ "" then (
334 0837 2             sp = ch$find ch(.string_dsc[len]-3,.string_dsc[ptr]+3, """);
335 0838 2             if .sp equ 0 then
336 0839 2                 return false;
337 0840 2                 build_descriptor(hex_dsc,.sp-.string_dsc[ptr]-3,.string_dsc[ptr]+3);
338 0841 2     ) else
```

```
339 0842 3 build_descriptor(hex_dsc,,string_dsc[len]-2,,string_dsc[ptr]+2);
340 0843 3 status = otsscvt_tz_l(hex_dsc,,longword,4,%b'1');
341 0844 3
342 0845 3 } else
343 0846 3
344 0847 3     ! We have a decimal number. Convert it and return the status.
345 0848 3
346 0849 3     status = otsscvt_tf_l(string_dsc,,longword,4,%b'11');
347 0850 3
348 0851 3 return .status;
349 0852 3
350 0853 3 end;
```

.PSECT SPLIT,NOWRT,NOEXE,2

58 25 00044 P.AAI: .ASCII \x1

.PSECT SCODES,NOWRT,2

RMS  
V04-000

RMS - Main Module for ANALYZE/RMS FILE  
ANL\$INTERNALIZE\_NUMBER - Convert String to Long

15-Sep-1984 23:52:21  
14-Sep-1984 11:52:58

VAX-11 Bliss-32 v4.0-742  
DISK\$VMSMASTER:[ANALYZE.SRC]RMS.B32:1

Page 17  
(6)

	04 00069	RET		
	03 DD 0006A	58:	PUSHL	#3
	04 DD 0006C		PUSHL	#4
08	AC DD 0006E		PUSHL	LONGWORD
	55 DD 00071		PUSHL	R5
00000000G 00	04 FB 00073		CALLS	#4. OTSSCVT_T1_L
	04 0007A		RET	
	50 D4 0007B	68:	CLRL	R0
	04 0007D		RET	

; 0830  
; 0849

; 0851  
; 0853

; Routine Size: 126 bytes. Routine Base: SCODES + 0143

```
352 0854 1 Isbttl 'ANL$CHECK_FLAGS = Check Flag Usage'  
353 0855 1 /*  
354 0856 1 Functional Description:  
355 0857 1 This routine is called to check the usage of flags in a flag  
356 0858 1 byte/word/longword. This routine is here for no better place.  
357 0859 1  
358 0860 1 Formal Parameters:  
359 0861 1 vbn  
360 0862 1 flags  
361 0863 1 flag_def  
362 0864 1  
363 0865 1  
364 0866 1  
365 0867 1  
366 0868 1 Implicit Inputs:  
367 0869 1 global data  
368 0870 1  
369 0871 1 Implicit Outputs:  
370 0872 1 global data  
371 0873 1  
372 0874 1 Returned Value:  
373 0875 1 none  
374 0876 1  
375 0877 1 Side Effects:  
376 0878 1  
377 0879 1 --  
378 0880 1  
379 0881 2 global routine anl$check_flags(vbn,flags,flag_def): novalue = begin  
380 0882 2  
381 0883 2 bind  
382 0884 2  
383 0885 2 flags_vector = flags: bitvector[],  
384 0886 2 flag_def_vector = .flag_def: vector[,long];  
385 0887 2  
386 0888 2  
387 0889 2 local  
388 0890 2  
389 0891 2  
390 0892 2 We will simply sit in a loop scanning the flag bits. If any flag is  
391 0893 2 set but undefined, we will issue an error message.  
392 0894 2  
393 0895 3 incr i from 0 to 31 do {  
394 0896 3  
395 0897 4 if .flags_vector[.i] then  
396 0898 4  
397 0899 4  
398 0900 4  
399 0901 4  
400 0902 4  
401 0903 4  
402 0904 4  
403 0905 4  
404 0906 4  
1 end;
```

20	08	AC	0004	00000		.ENTRY	ANL\$CHECK_FLAGS, Save R2	0882
	0C	BC	52	D4	00002	CLRL	I	0895
			52	E1	00004	BBC	I, FLAGS_VECTOR, 38	0896
			52	D1	00009	CMPL	I, @FLAG_DEF	0897
	50	0C	0A	1A	0000D	BGTRU	28	0898
		BC	04	A0	00014	MOVAL	@FLAG_DEF[I], R0	0901
			10	12	00017	TSTL	4(R0)	
			52	DD	00019	BNEQ	38	
			04	AC	0001B	PUSHL	I	
	0000G	CF	00000000G	8F	0001E	PUSHL	VBN	
			03	FB	00024	PUSHL	#ANL\$RMSS_FLAGERROR	
			52	D6	00029	CALLS	#3, ANL\$FORMAT_ERROR	0895
		1F	52	D1	0002B	INCL	I	
			D4	1B	0002E	CMPL	I, #31	
			04	00030		BLEQU	18	
						RET		0906

; Routine Size: 49 bytes, Routine Base: \$CODES + 01C1

```
406 0907 1 Xsbttl 'ANL$PREPARE_QUOTED_STRING - Prepare a Quoted String'
407 0908 1 ++
408 0909 1 Functional Description:
409 0910 1 This routine is called to prepare a quoted string for inclusion in
410 0911 1 an FDL specification, or perhaps in a formatted message. Preparing
411 0912 1 the string includes stripping trailing whitespace, doubling any
412 0913 1 quotation marks, and enclosing it in quotation marks.
413 0914 1
414 0915 1 Formal Parameters:
415 0916 1      input_dsc      Descriptor of buffer with input string.
416 0917 1      output_dsc     Descriptor of buffer to receive output string.
417 0918 1                  The length is set correctly.
418 0919 1
419 0920 1 Implicit Inputs:
420 0921 1      global data
421 0922 1
422 0923 1 Implicit Outputs:
423 0924 1      global data
424 0925 1
425 0926 1 Returned Value:
426 0927 1      none
427 0928 1
428 0929 1 Side Effects:
429 0930 1
430 0931 1      --
431 0932 1
432 0933 1
433 0934 1 global routine anl$prepare_quoted_string(input_dsc: ref descriptor,
434 0935 1                                output_dsc: ref descriptor):
435 0936 2      novalue = begin
436 0937 2
437 0938 2 bind
438 0939 2      input_vector = .input_dsc[ptr]: vector[,byte].
439 0940 2      output_vector = .output_dsc[ptr]: vector[,byte];
440 0941 2
441 0942 2 local
442 0943 2      i: signed long,
443 0944 2      trimmed_length: long;
444 0945 2
445 0946 2
446 0947 2      ! Begin by scanning the input string from the end in order to eliminate
447 0948 2      any trailing whitespace. We actually eliminate all control characters
448 0949 2      so that we'll catch NULs too.
449 0950 2
450 0951 2      i = (decr i from .input_dsc[len]-1 to 0, do
451 0952 2          if .input_vector[i] gtru ',' then exitloop .i);
452 0953 2      trimmed_length = .i + 1;
453 0954 2
454 0955 2      ! Put the opening quotation mark in the output buffer.
455 0956 2
456 0957 2      output_vector[0] = '"';
457 0958 2      output_dsc[len] = 1;
458 0959 2
459 0960 2      ! Scan the input string from the beginning, moving each character into the
460 0961 2      ! output buffer. Quotation marks must be doubled.
461 0962 2
462 0963 3      incr i from 0 to .trimmed_length-1 do {
```

```

: 463 0964 4      if .input_vector[i] equu "" then {
: 464 0965 4          output_vector[.output_dsc[len]] = "";
: 465 0966 4          increment(output_dsc[len]);
: 466 0967 3      );
: 467 0968 3      output_vector[.output_dsc[len]] = .input_vector[i];
: 468 0969 3      increment(output_dsc[len]);
: 469 0970 2      );
: 470 0971 2      ! Add the closing quotation mark to the output buffer.
: 471 0972 2      output_vector[.output_dsc[len]] = "";
: 472 0973 2      increment(output_dsc[len]);
: 473 0974 2      return;
: 474 0975 2
: 475 0976 2
: 476 0977 2
: 477 0978 2
: 478 0979 1 end;

```

								.ENTRY ANL\$PREPARE_QUOTED_STRING, Save R2,R3,R4,R5 : 0934
52	04	AC 7D 00002				MOVQ	INPUT_DSC, R2 : 0939	
54	04	A3 D0 00006				MOVL	4(R3), R4 : 0940	
50		62 3C 0000A				MOVZWL	(R2), I : 0951	
		07 11 0000D				BRB	2\$	
20	04 B240	91 0000F	1\$:			CMPB	@4(R2)[I], #32 : 0952	
		04 1A 00014				BGTRU	3\$	
		50 D7 00016	2\$:			DECL	I	
		F5 11 00018				BRB	1\$	
		50 D6 0001A	3\$:			INCL	TRIMMED LENGTH : 0953	
64		22 90 0001C				MOVBL	#34, (R4) : 0957	
63		01 B0 0001F				MOVWL	#1, (R3) : 0958	
51		01 CE 00022				MNEGL	#1, I : 0963	
		1B 11 00025				BRB	6\$	
22	04 B241	91 00027	4\$:			CMPB	@4(R2)[I], #34 : 0964	
		09 12 0002C				BNEQ	5\$	
55		63 3C 0002E				MOVZWL	(R3), R5 : 0965	
6544		22 90 00031				MOVBL	#34, (R5)[R4] : 0966	
		63 B6 00035				INCW	(R3) : 0968	
55		63 3C 00037	5\$:			MOVZWL	(R3), R5 : 0969	
6544	04 B241	90 0003A				MOVBL	@4(R2)[I], (R5)[R4] : 0970	
		63 B6 00040				INCW	(R3) : 0971	
E1	51	50 F2 00042	6\$:			A0BLSS	TRIMMED LENGTH, I, 4\$ : 0972	
	50	63 3C 00046				MOVZWL	(R3), R0 : 0973	
6044		22 90 00049				MOVBL	#34, (R0)[R4] : 0974	
		63 B6 0004D				INCW	(R3) : 0975	
		04 0004F				RET	: 0979	

: Routine Size: 80 bytes. Routine Base: \$CODE\$ + 01F2

: 479 0980 1  
: 480 0981 0 end eludom

RMS  
V04-000

RMS - Main Module for ANALYZE/RMS\_FILE  
ANL\$PREPARE\_QUOTED\_STRING - Prepare a Quoted St

B 8  
15-Sep-1984 23:52:21  
14-Sep-1984 11:52:58

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[ANALYZ.SRC]RMS.B32;1

Page 22  
(8)

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name

Bytes

Attributes

\$GLOBALS	8	NOVEC, WRT, RD	NOEXE,NOSHR, LCL, REL,	CON,NOPIC,ALIGN(2)
\$SPLITS	70	NOVEC,NOWRT, RD	NOEXE,NOSHR, LCL, REL,	CON,NOPIC,ALIGN(2)
\$CODES	578	NOVEC,NOWRT, RD	EXE,NOSHR, LCL, REL,	CON,NOPIC,ALIGN(2)

Library Statistics

file

----- Symbols -----			Pages Mapped	Processing Time
Total	Loaded	Percent		
18619	17	0	1000	00:01.8

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:RMS/OBJ=OBJ\$:RMS MSRC\$:RMS/UPDATE=(ENH\$:RMS)

Size: 578 code + 78 data bytes  
Run Time: 00:16.8  
Elapsed Time: 00:54.3  
Lines/CPU Min: 3497  
Lexemes/CPU-Min: 16260  
Memory Used: 170 pages  
Compilation Complete

0007 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY